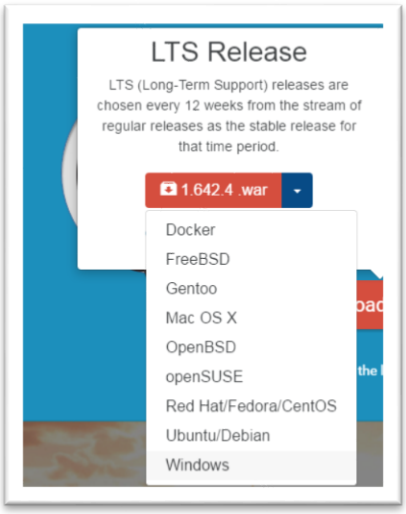
How do I use Polyspace Bug Finder with Jenkins?

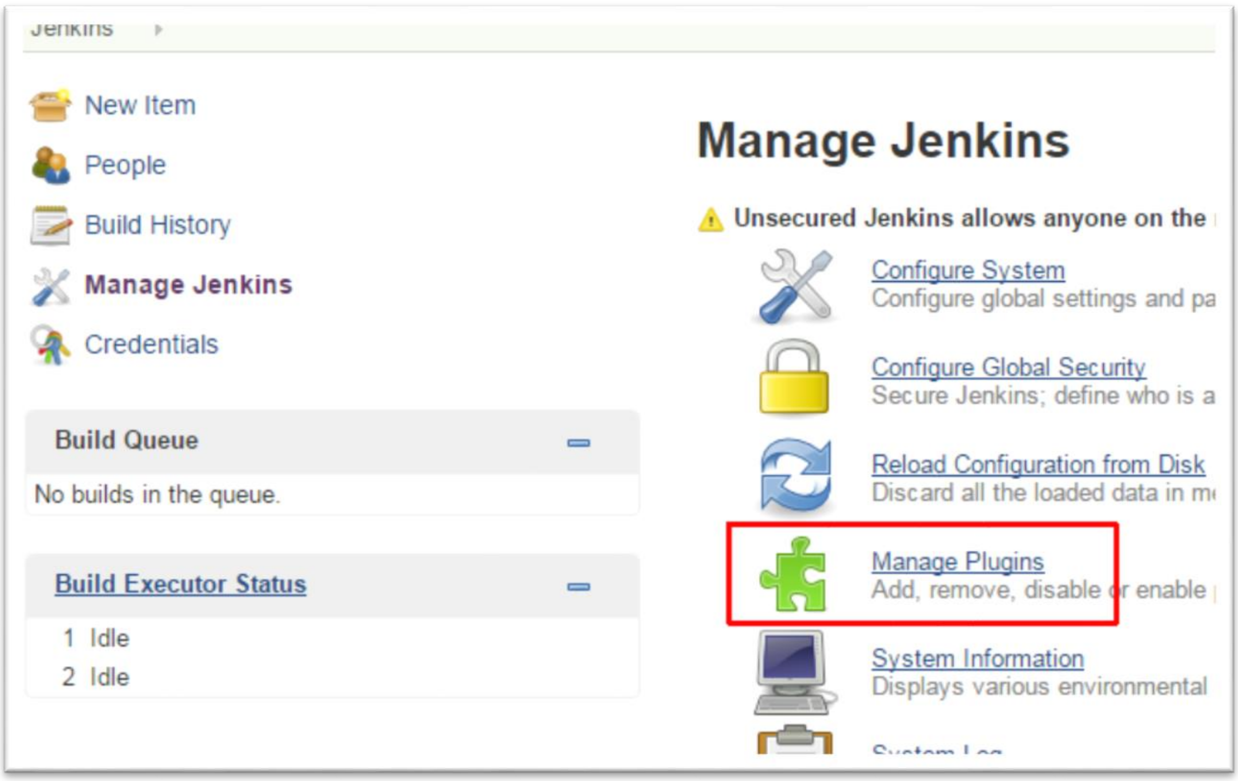
What do you need?

This Jenkins integration can be used with Polyspace Bug Finder since R2013b under a Windows system (note that it would be easy to adapt the bat scripts in order to make them work under Linux or Mac).
In this demonstration we will use the MinGW environment under Windows, and use g++ to compile C++ example files.
You need also to install Jenkins from www.jenkins-ci.org. Click the download button and then choose your platform:


LTS Release
LTS (Long-Term Support) releases are chosen every 12 weeks from the stream of regular releases as the stable release for that time period.
1.642.4 .war
Docker
FreeBSD
Gentoo
Mac OS X
OpenBSD
openSUSE
Red Hat/Fedora/CentOS
Ubuntu/Debian
Windows

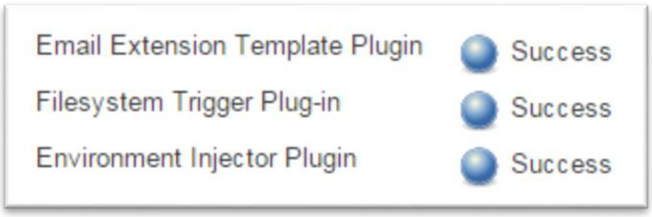
Under Windows, a zip file containing the file jenkins.msi file will be downloaded.

You will also need to install additional plugins by going to manage plugins directly in Jenkins:


Jenkins
New Item
People
Build History
Manage Jenkins
Credentials
Build Queue
No builds in the queue.
Build Executor Status
1 Idle
2 Idle
Manage Jenkins
Unsecured Jenkins allows anyone on the
Configure System
Configure global settings and pa
Configure Global Security
Secure Jenkins; define who is a
Reload Configuration from Disk
Discard all the loaded data in m
Manage Plugins
Add, remove, disable or enable
System Information
Displays various environmental

Install:

1. Email extension template plugin, to send E-mail when a build has been successful or failed.
2. Filesystem Trigger Plug-in, to start a Jenkins job when a file has been modified.
3. Environment Injector Plugin, to add environment variable inside Jenkins.



Email Extension Template Plugin	● Success
Filesystem Trigger Plug-in	● Success
Environment Injector Plugin	● Success

What is the content of the zip file?

Unzip PolyspaceForJenkins.zip in a Polyspace Tools folder like `C:\Polyspace\tools\Jenkins` (the proposed folder). We will refer to <ps_jenkins> folder in the following.

Contents of the zip file:

- A `<ps_jenkins>\Workspace` sub-folder.
- A `<ps_jenkins>\Workspace\Tools` folder containing some bat script files.
- A `<ps_jenkins>\Workspace\Docs` folder containing the current documentation in a pdf format.
- A `<ps_jenkins>\Workspace\Tools\lib` containing a Perl script
- A `<ps_jenkins>\Workspace\xml` containing xml Jenkins configuration and Monopoly job configuration
- A `<ps_jenkins>\Monopoly` sub-folder that contains some C++ files that will be used to demonstrate usage of Polyspace Bug Finder in a continuous integration system.

What workflow does it demonstrate?

The workflow is quite simple: as soon as you estimate that you need to launch a new run of Polyspace Bug Finder on a project periodically (every week for instance) or regularly after particular events (commit, complete build, etc.), Jenkins can run a job for you and warn as soon as it has been finished.
The proposed workflow described below is well adapted when you have your complete application and you want to analyse it on a regular basis or after some changes.

The workflow in Jenkins is the following:

- Wait for an event saying that you are ready to analyse your source code. The event caught by Jenkins will start the following jobs in a sequential way.
    a. *polyspace-configure* monitors the compilation of your complete project (a build "all") and creates a project file (an options file).
    b. Then Polyspace Bug Finder analysis starts on the built files.
    c. Polyspace "*differ*", at end of the analysis, executes a difference between previous job results and current job results.
    d. One script parses the two last log files and differ txt file to check the difference with a previous run.
    e. At last, an E-mail will be sent with the difference.
- From E-mail received, a link is proposed to open results.

Note that as soon as one of the previous steps fails and the complete job stops you receive an E-mail with a "failed" status along with the log of job.

How do I configure Jenkins?

Default Jenkins configuration

You need administration rights.

1. After installing Jenkins and the three additional plugins (using http://localhost:8080/pluginManager/), stop Jenkins service in Windows.
2. Then copy the following xml files in Jenkins install folder:
   a. Then , you have to update `husdson.tasks.Mailer.xml` and `hudson.plugins.emailext.ExtendedEmailPublisher` using notepad++ or with any xml editor before copying it.
      They contain both `smtp` and E-mail suffix addresses that should be adapted to your own environment.

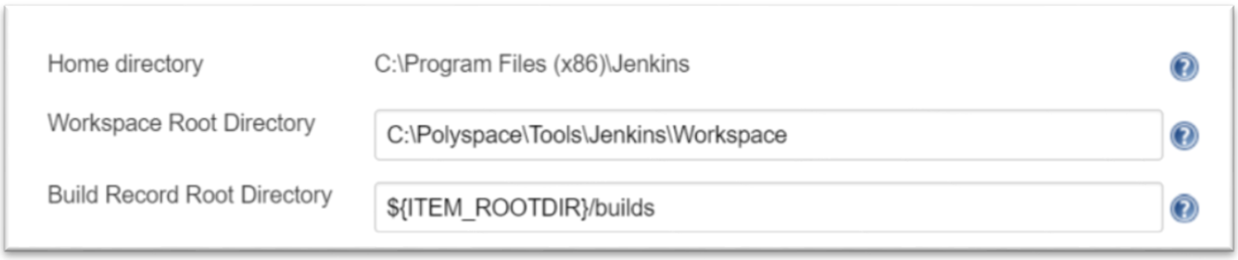
   b. Then copy all the following files in the default install folder of Jenkins `c:\Program Files (x86)\Jenkins`. I will refer it as `$JENKINS_HOME` in the following.
   - `<ps_jenkins>\Workspace\xml\jenkins.xml,`
   - `<ps_jenkins>\Workspace \xml\hudson.tasks.Mailer.xml,`
   - `<ps_jenkins>\Workspace\xml\hudson.plugins.emailext.ExtendedEmailPublisher,`
   - `<ps_jenkins>\Workspace\xml\jenkins.model.JenkinsLocationConfiguration.xml,` and
   - `<ps_jenkins>\Workspace\xml\config.xml`

   All these xml files contain the Jenkins configuration used.
3. You can afterward restart Jenkins from Windows Services.

Set Jenkins configuration from scratch

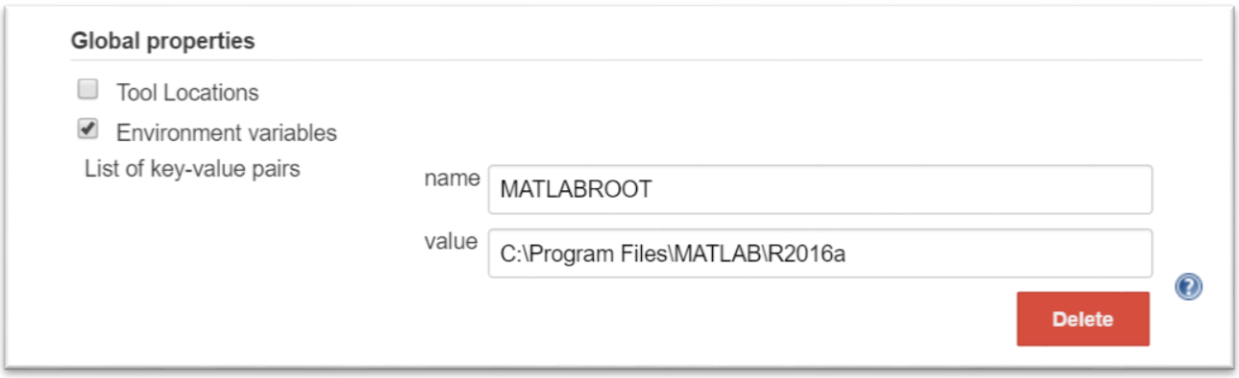
To set the configuration by hand, type in your preferred browser (like chrome or IE) the following command: http://localhost:8080/configure and then change the following parameters:

- Change "Workspace Root Directory"



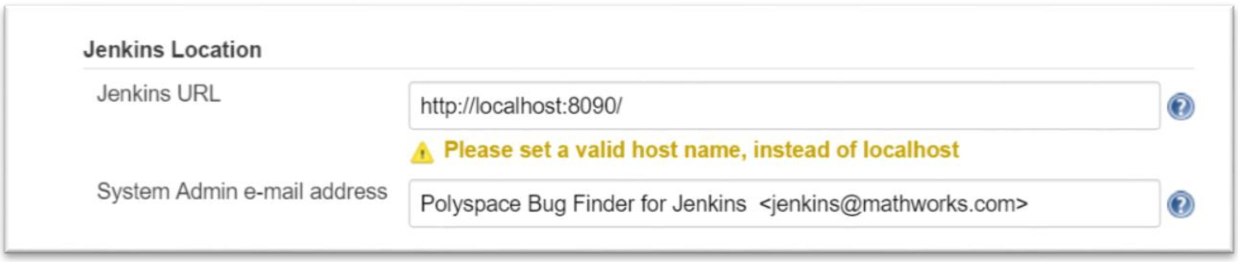
Home directory	C:\Program Files (x86)\Jenkins
Workspace Root Directory	C:\Polyspace\Tools\Jenkins\Workspace
Build Record Root Directory	${ITEM_ROOTDIR}/builds

- Add MATLABROOT environment variable:


Global properties
- [ ] Tool Locations
- [x] Environment variables

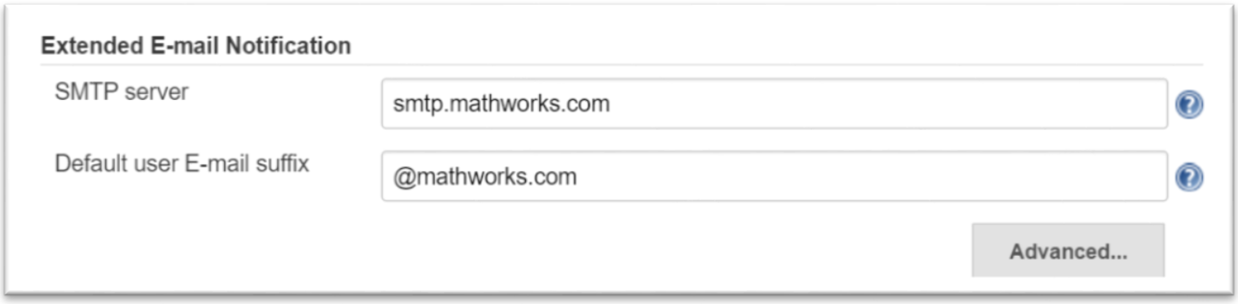
List of key-value pairs
name: MATLABROOT
value: C:\Program Files\MATLAB\R2016a

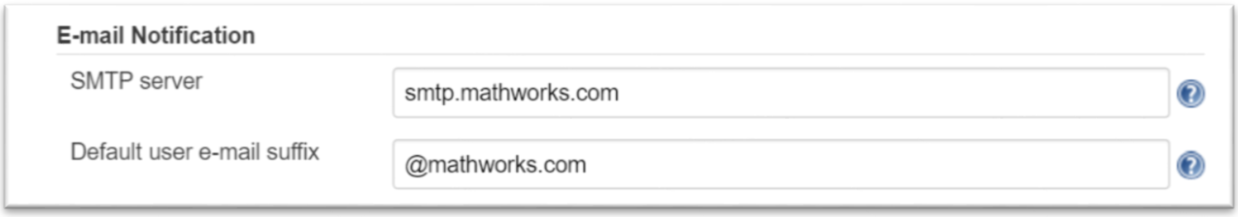
- Change Jenkins URL to port **8090**:


Jenkins Location
Jenkins URL: http://localhost:8090/
Please set a valid host name, instead of localhost
System Admin e-mail address: Polyspace Bug Finder for Jenkins <jenkins@mathworks.com>

Then Jenkins will be accessible with command http://localhost:8090/. Jenkins URL port has been changed to 8090 in order to prevent a conflict with Polyspace web metric server which can be accessible by default with URL port 8080.

- Set you SMTP server in Extended and in Standard E-mail Notification:


Extended E-mail Notification
SMTP server: smtp.mathworks.com
Default user E-mail suffix: @mathworks.com


E-mail Notification
SMTP server: smtp.mathworks.com
Default user e-mail suffix: @mathworks.com

How do I install the Monopoly item?

Set default Job configuration

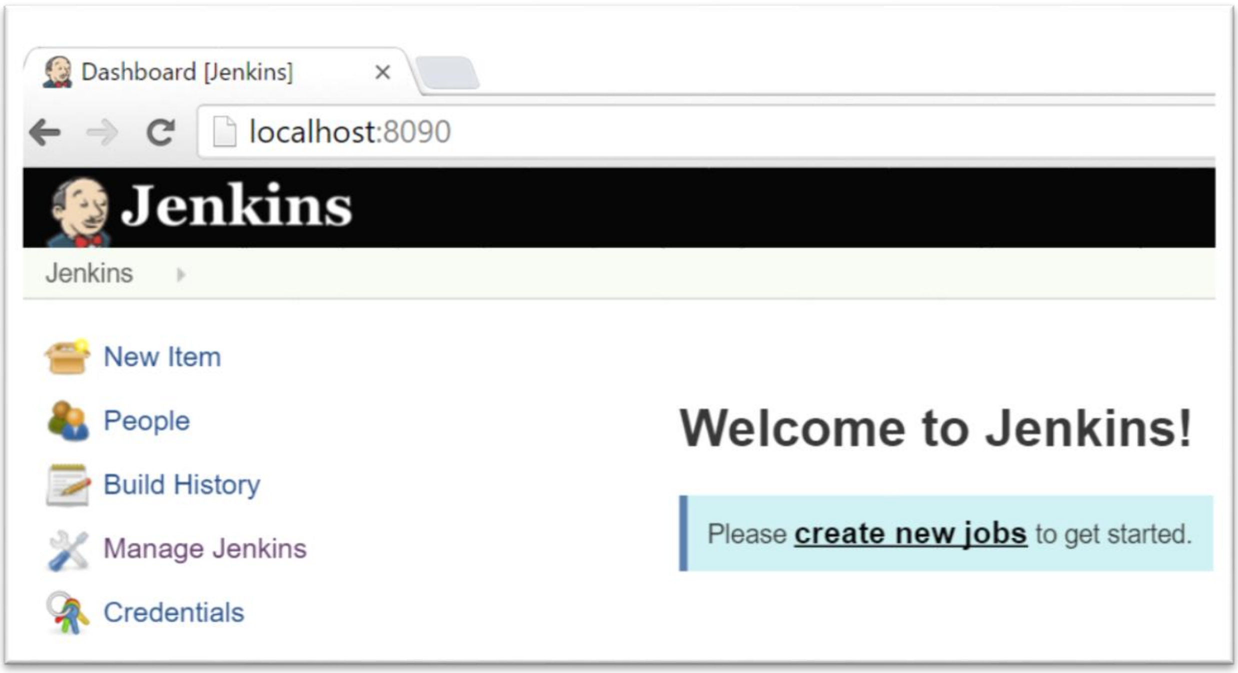
`<ps_jenkins>\Workspace\xml\monopoly.xml` contains the Jenkins configuration used for the Monopoly demonstration example.

1. Stop Jenkins service in Windows.
2. Create a sub-folder `$JENKINS_HOME\jobs\Monopoly`

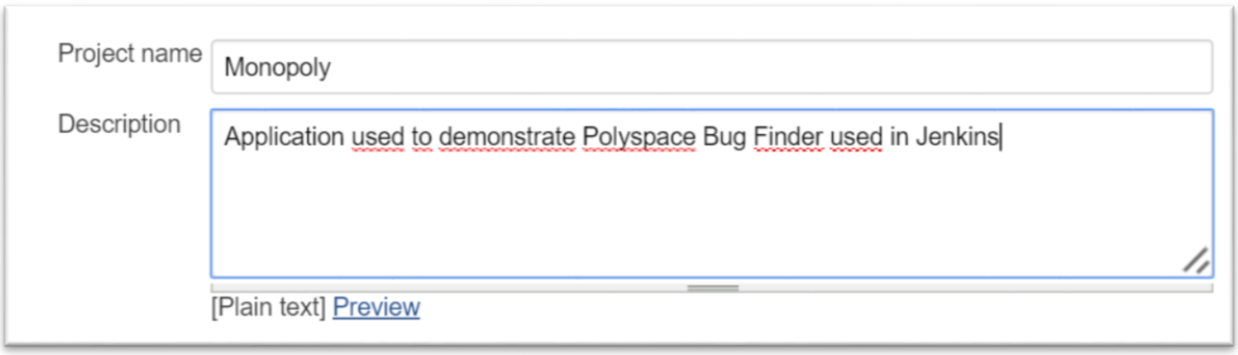
3. Then copy the following xml file `<ps_jenkins>\Workspace\xml\monopoly.xml` in folder `$JENKINS_HOME\jobs\Monopoly`, and
4. Then rename it as `config.xml`
5. At last, restart Jenkins service in Windows

Set Job Configuration from scratch

Create one new item from scratch with “New Item”: http://localhost:8090/view/All/newJob



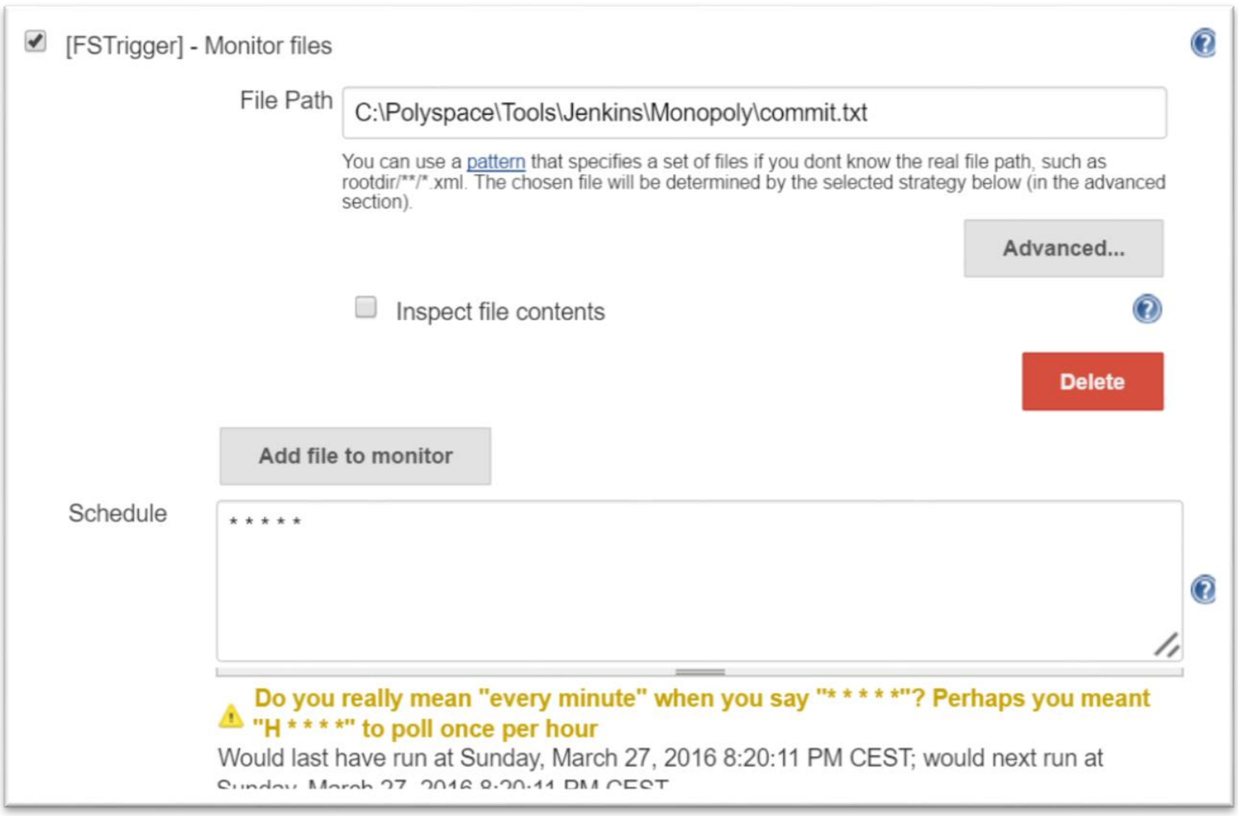
- Add Project name



Project name: Monopoly

Description: Application used to demonstrate Polyspace Bug Finder used in Jenkins

- Build Triggers



☑ [FSTrigger] - Monitor files

File Path: C:\Polyspace\Tools\Jenkins\Monopoly\commit.txt

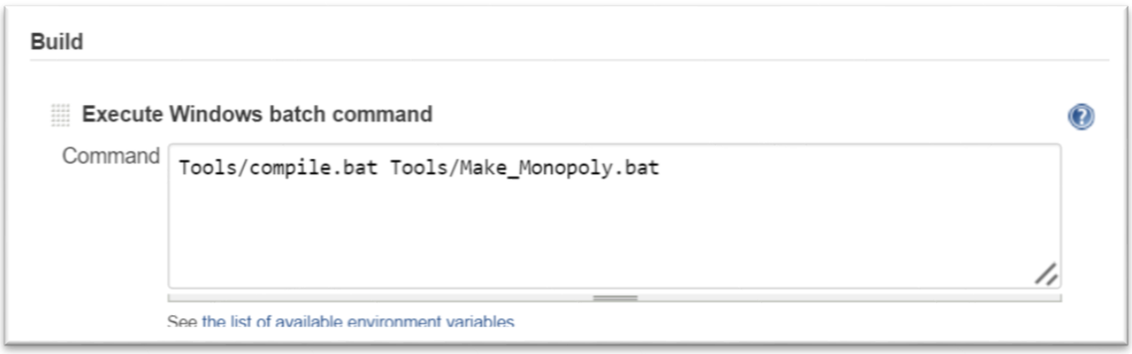
You can use a pattern that specifies a set of files if you dont know the real file path, such as rootdir/**/*.xml. The chosen file will be determined by the selected strategy below (in the advanced section).

Schedule: `* * * * *`

Do you really mean "every minute" when you say "* * * * *"? Perhaps you meant "H * * * *" to poll once per hour

Would last have run at Sunday, March 27, 2016 8:20:11 PM CEST; would next run at

- Create Builds
  - First, polyspace-configure monitors the build command and creates a configuration file



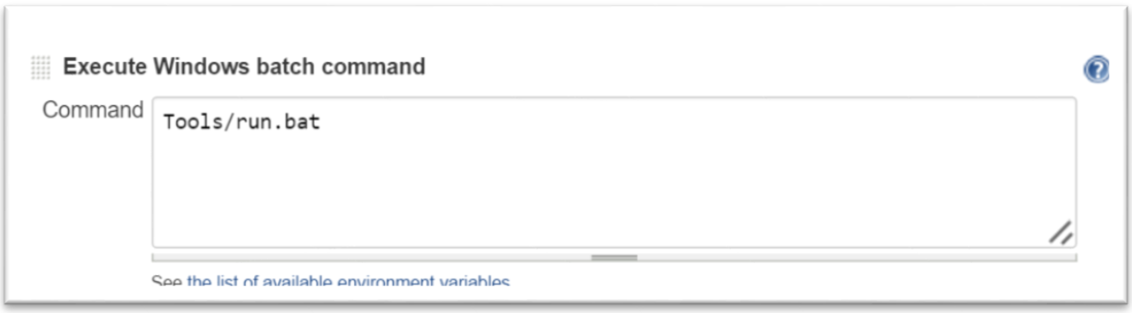
Build

Execute Windows batch command

Command: `Tools/compile.bat Tools/Make_Monopoly.bat`

`<ps_jenkins>\Workspace\Tools\Make_Monopoly.bat` is the parameter of the Windows Batch file `<ps_jenkins>\Workspace\Tools\compile.bat`. It is the build command of the Monopoly application: it is a very simple build command allowing to compile with g++, two C++ files located in `<ps_jenkins>\Monopoly`.

  - Then, a bug finder analysis starts as soon as files have compiled successfully



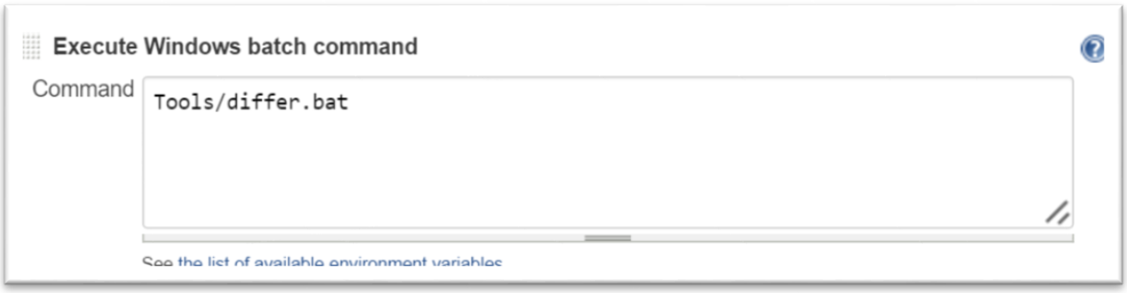
Execute Windows batch command

Command

```
Tools/run.bat
```

See the list of available environment variables

- Then, Polyspace calculates difference between previous analysis and current analysis



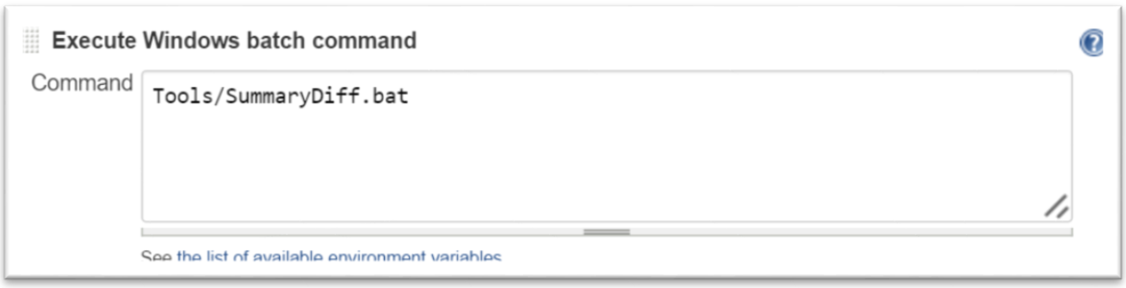
Execute Windows batch command

Command

```
Tools/differ.bat
```

See the list of available environment variables

- At last, a Perl script prepares an HTML summary that will be part of an E-mail giving numbers of defects and coding rule violations.



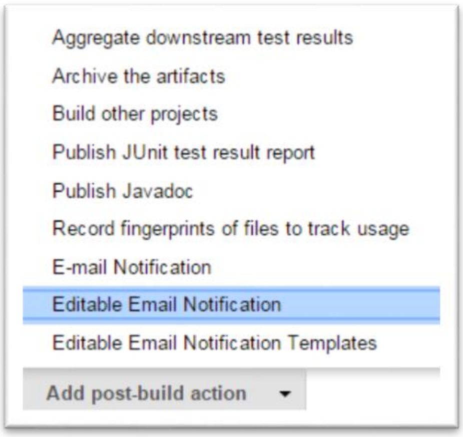
Execute Windows batch command

Command

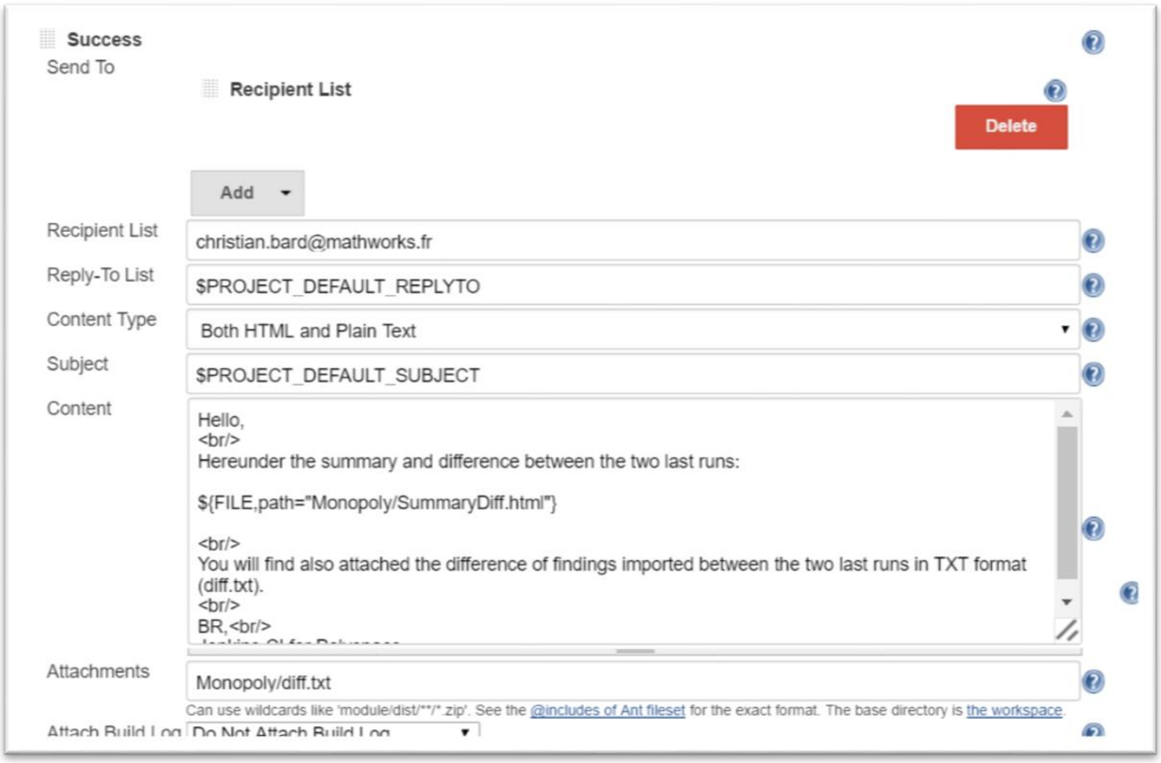
```
Tools/SummaryDiff.bat
```

See the list of available environment variables

- Create a post-build action (Select Editable Email Notification):


Aggregate downstream test results
Archive the artifacts
Build other projects
Publish JUnit test result report
Publish Javadoc
Record fingerprints of files to track usage
E-mail Notification
Editable Email Notification
Editable Email Notification Templates
Add post-build action

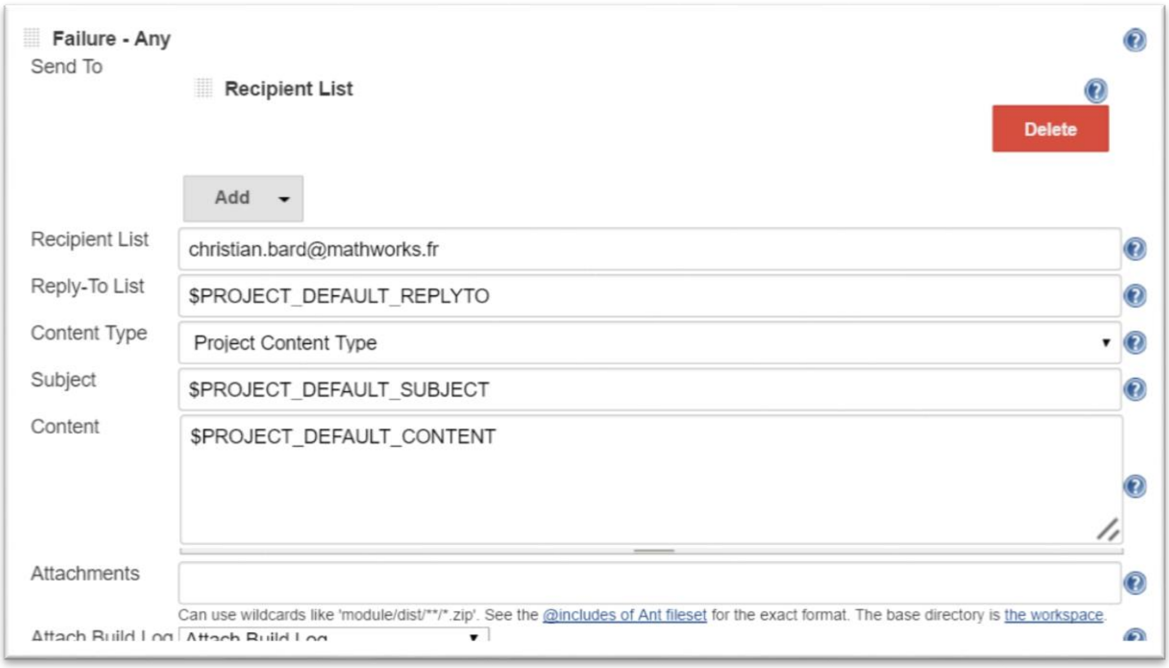
- Then Trigger “`Success`” and attach HTML file and diff.txt:


Success
Send To
Recipient List
Delete
Add
Recipient List: christian.bard@mathworks.fr
Reply-To List: $PROJECT_DEFAULT_REPLYTO
Content Type: Both HTML and Plain Text
Subject: $PROJECT_DEFAULT_SUBJECT
Content:
```
Hello,
<br/>
Hereunder the summary and difference between the two last runs:

${FILE,path="Monopoly/SummaryDiff.html"}

<br/>
You will find also attached the difference of findings imported between the two last runs in TXT format
(diff.txt).
<br/>
BR,<br/>
```
Attachments: Monopoly/diff.txt
Can use wildcards like 'module/dist/**/*.zip'. See the @includes of Ant fileset for the exact format. The base directory is the workspace.
Attach Build Log: Do Not Attach Build Log

- Also Trigger “`Failure - Any`” and attach build log:


Failure - Any
Send To
Recipient List
Delete
Add
Recipient List: christian.bard@mathworks.fr
Reply-To List: $PROJECT_DEFAULT_REPLYTO
Content Type: Project Content Type
Subject: $PROJECT_DEFAULT_SUBJECT
Content: $PROJECT_DEFAULT_CONTENT
Attachments:
Can use wildcards like 'module/dist/**/*.zip'. See the @includes of Ant fileset for the exact format. The base directory is the workspace.
Attach Build Log: Attach Build Log

Scripts used for a Monopoly job

This paragraph describes content of scripts located in the Jenkins `<ps_jenkins>\Workspace\Tools` and `<ps_jenkins>\Workspace\Tools\lib` sub-folders.

compile.bat

Allows to run polyspace-configure on the build command and to generate a textual configuration file used later to run Polyspace Bug Finder.

Inputs: The build command as argument (`%MBUILD%`).
Jenkins environment variables: `%JOB_NAME%` (name of the job), `%WORKSPACE%` (Localization of the workspace), `%MROOT%` (localization of MATLAB folder).
Outputs: configuration file located in project sub-folder as `%WORKSPACE%\%JOB_NAME%\%JOB_NAME%.opts`.

run.bat

Allows to execute Polyspace Bug Finder after project build. It uses configuration file generated from previous script and also an additional configuration file that would represent some additional options like a list of specific defects and coding rules.
Named `<ps_jenkins>\Workspace\Tools\target_%JOB_NAME%.opts` it includes also some specific settings associated to the project itself and to the compiler, options not caught by `polyspace-configure`.
Most of the time, this file contains some additional options like the list of coding rules you want to check w/ the project and some settings that are generic to the project like some intrinsic defines associated with the compiler, etc.

Global inputs: `%WORKSPACE%\Tools\target_%JOB_NAME%.opts` and `%WORKSPACE%\%JOB_NAME%\%JOB_NAME%.opts`.
Additional Jenkins environment variables: `%BUILD_ID%, %BUILD_NUMBER%` (#ID job in Jenkins) and `%JOB_NAME%` (name of the project).
Outputs: results folder localized in `%WORSPACE%\%JOB_NAME%\R_BF_%BUILD_NUMBER%` (`<ps_jenkins>\Workspace\Monopoly\R_BF_#ID` for the Monopoly project).

differ.bat

Allows to compute the difference between previous run and current run using polyspace-comments-import.exe.
It generates a text file `diff.txt`. The format of `diff.txt` is something like below:

```
Operation completed:
--------------------
<List of specific defects and coding rules> added or removed in the current run>

Diff statistics for Defect results:
-----------------------------------
| Number of unmatched commented results                              : #value
| Number of unmatched uncommented results                            : #value
| Number of results comments not imported because a more recent one was found: #value
| Number of partially imported results comments (due to a result change)     : #value
| Number of fully imported results comments                          : #value
| Number of matched results without comment                          : #value
| Number of matched results with a comment coming from source code annotation: #value
| Total number of results                                            : #value

Diff statistics for <coding rules> results:
--------------------------------------
| Number of unmatched commented results                              : #value
| Number of unmatched uncommented results                            : #value
| Number of results comments not imported because a more recent one was found: #value
| Number of partially imported results comments (due to a result change)     : #value
| Number of fully imported results comments                          : #value
| Number of matched results without comment                          : #value
| Number of matched results with a comment coming from source code annotation: #value
| Total number of results                                            : #value
```

Global Inputs: The previous results folder `%WORSPACE%\%JOB_NAME%\R_BF_%BUILD_NUMBER%-1` and the current results folder of the job `%WORSPACE%\%JOB_NAME%\R_BF_%BUILD_NUMBER%`.
Outputs: the text file `%WORSPACE%\%JOB_NAME%\diff.txt`.

SummaryDiff.bat

This script executes the Perl script `%WORSPACE%\Tools\lib\SummaryDiff.pl`. It creates an HTML file giving the number of defects, coding rules and a link to the last two results folders.
Format is the following:

```
Polyspace Bug Finder Results Summary

Summary of results found in <workspace\ProjectName> sub-folder
```

name	# of defects	Lang	Files	Line	Time (s)	Coding Rules	Link to open results
R_BF_50	#value1	CPP	12	679	30.17	#value1	Open results
R_BF_51	#value2	CPP	12	679	31.16	#value2	Open results
# to review	value					value	

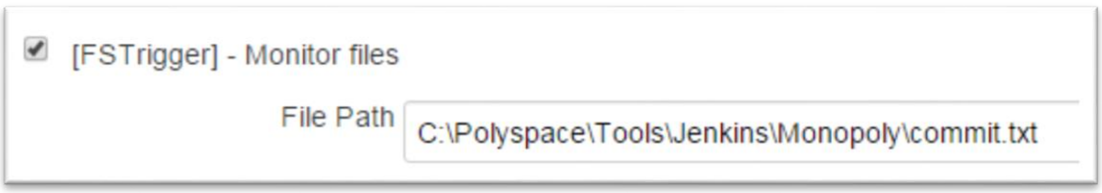
```
Results summary generated at <time and date>
```

Inputs: name of project as first argument, the previous results folder and the current results folder of the job.
Outputs: `%WORSPACE%\%JOB_NAME%\SummaryDiff.html`

How do I run a new job ID?

Run a new job with `commit_Monopoly.bat`

The « Monopoly » job monitors a possible update of the file `commit.txt` every minute:



☑ [FSTrigger] - Monitor files

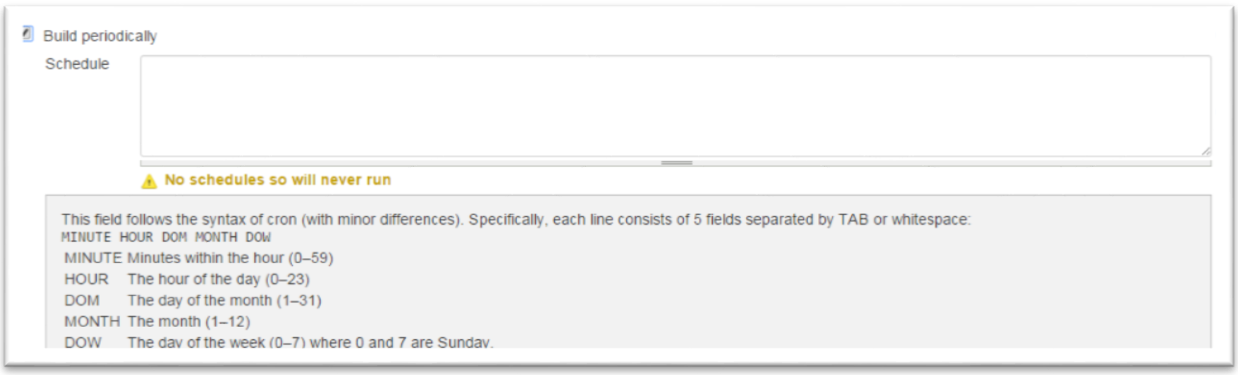
File Path: C:\Polyspace\Tools\Jenkins\Monopoly\commit.txt

To start a new job, you just have to execute `<ps_jenkins>\Workspace\Tools\commit_Monopoly.bat`.

The script just does an update of `commit.txt`. Then, one minute later, a new Monopoly job starts. At the end, an E-mail with a "success" or "failed" status from `jenkins@mathworks.com` is sent to user (see § Set Jenkins configuration from scratch)

How do I set a periodic run?

To run a job periodically, you just need to tick "Build periodically" and add a schedule.


Build periodically
Schedule

No schedules so will never run

This field follows the syntax of cron (with minor differences). Specifically, each line consists of 5 fields separated by TAB or whitespace:
```
MINUTE HOUR DOM MONTH DOW
```
MINUTE	Minutes within the hour (0–59)
HOUR	The hour of the day (0–23)
DOM	The day of the month (1–31)
MONTH	The month (1–12)
DOW	The day of the week (0–7) where 0 and 7 are Sunday.

How do I add a new job in Jenkins?

You need to create a new item in Jenkins with a new Name, and follows paragraph Set Job Configuration from scratch. Then you just have to create two new files: (1) a new file `target_<new Name>.opts` (see run.bat) and also propose a way to build your project like `Make_Monopoly.bat` (see compile.bat).

Note that you may copy `<ps_jenkins>\Workspace\xml\monopoly.xml` in Jenkins Jobs sub-folder, create a new folder with name of your job and rename it as config.xml (see paragraph Set default Job configuration).

Limitations and troubleshooting[1]

- The first time you run a job, the script SummaryDiff.bat will fail since no check is done on the existence of a previous analysis.
- Actually the Jenkins Working space has to be shared on same disk (same drive and location) user receiving E-mails.
  - You need to adapt the SummaryDiff.pl script to take into account a shared folder on an internal network
  - You will need to add a Build script that does the copy of the results folder to another location on the network in order that user receiving E-mail would open results.
- Opening results from E-mail is only possible after associating `ps_results.psbf` to `<MATLABROOT>\polyspace\bin\polyspace-bug-finder.exe`.
- Multi-users is not managed
- SummaryDiff.pl needs some adaptation for a possible usage under another OS.

[1] Any problems and feedback, please send an E-mail to christian.bard@mathworks.com